



**《程序设计原理》实验报告**

**——更大的整数**

**学 号 3022244290**

**姓 名 陈秋澄**

**学 院 智能与计算学部**

**年 级 22级**

**任课教师 李雪威**

**2022年12月10日**

# 实验目的

由于C++中，整数在使用时无法覆盖所有需要“数字”的情况。为解决这个问题，如果我们将整数看作是“类”，以面向对象的角度去完善它，即以接口的方式实现整数，我们就可以获得通用性更强的整数类“MyInteger”，而设计更强的整数类，就可以扩大整数的有效范围。

# 二．实验内容

1. **设计思路**

计算方式为“倒着算”：因为算数的位数和数组是倒过来的，所以先把数据倒过来，再依次计算。

首先，设计构造函数：用于默认构造很熟构造函数且默认值为0的MyInteger()，用于从字符符串构造为大整数的MyInteger(string& num)，和用于从整型构造为大整数的MyInteger(int num)。

其次，运用string getnum()等成员函数确保程序正确运行。

1. **代码实现**

class MyInteger {

private:

    int n[10000], len;

public:

    MyInteger() { //默认构造很熟构造函数, 默认值为0

        memset(n, 0, sizeof(n));

        len = 0;

    }

    MyInteger(int num) { //构造函数，从整型构造为大整数

        memset(n, 0, sizeof(n));

        len = 0;

        for (int i = 0; num != 0; i++) {

            n[i] = num % 10;

            num /= 10;

            len++;

        }

    }

    MyInteger(string num) {

        memset(n, 0, sizeof(n));

        len = 0;

        string a; int count = 0;

        while (num[count] == '0')

            count++;

        while (num[count] != '\0')

            a.push\_back(num[count++]);

        reverse(a.begin(), a.end());

        for (int i = 0; i < strlen(a.c\_str()); i++){

            int temp; string temp1;

            temp1.push\_back(a[i]);

            sscanf(temp1.c\_str(), "%d", &temp);

            n[i] = temp;

            len++;

        }

    }

    ~MyInteger() {} //析构函数

    string getnum() { //  返回表示大整数的字符字符串，去除前导零。

        if (len == 0)return "0";

        string a;

        for (int i = 0; i < len; i++)

            a.push\_back(n[i] + 48);

        reverse(a.begin(), a.end());

        return a;

    }

MyInteger operator+(const MyInteger& b) { //大整数加法

        MyInteger c;

        int carry = 0;

        for (int i = 0; i < len || i < b.len; ++i){

            int temp = n[i] + b.n[i] + carry;

            carry = temp / 10;

            c.n[c.len++] = temp % 10;

        }

        if (carry != 0)

            c.n[c.len++] = carry;

        return c;

    }MyInteger operator-(const MyInteger& b) { //大整数减法

        MyInteger c;

        for (int i = 0; i < len || i < b.len; ++i){

            if (n[i] < b.n[i]) {

                n[i + 1]--;

                n[i] += 10;

            }

            c.n[c.len++] = n[i] - b.n[i];

        }

        while (c.len - 1 >= 1 && c.n[c.len - 1] == 0)

            c.len--;

        return c;

    }

    MyInteger operator\*(const MyInteger& b) { //大整数乘法

        if (b.len == 0)return b;

        MyInteger c, temp;

        MyInteger tb = b;

        for (int i = 0; i < len; i++) {

            for (int j = 0; j < tb.len; j++) {

                c.n[i + j] += n[i] \* tb.n[j];

            }

        }

        int l = len + tb.len;

        for (int i = 0; i < l; i++){

            c.n[i + 1] += c.n[i] / 10;

            c.n[i] %= 10;

        }

        if (c.n[l] > 0)

            l++;

        while (c.n[l - 1] >= 10){

            c.n[l] = c.n[l - 1] / 10;

            c.n[l - 1] %= 10;

            l++;

        }

        while (c.n[l - 1] == 0 && l > 1) l--;

        for (int i = 0; i < l; i++)

            temp.n[i] = c.n[i];

        temp.len = l;

        return temp;

    }

};

1. **结果展示**

****

# 三．实验中遇到的问题和解决办法

问题：方法选择不当，写出时间复杂度过高或空间复杂度过高的代码指针相关知识掌握得不扎实，还出现内存泄漏的情况。

解决办法：复习指针、析构函数等知识，综合指针、栈等知识，多次进行代码的编写和测试。